PAITS: A Provenance Collection Framework for Internet of Things (IoT) Systems

Ebelechukwu Nwafor, Andre Campbell, David Hill, and Gedare Bloom
Department of Electrical Engineering and Computer Science
Howard University, Washington, DC 20059
Email: ebelechukwu.nwafor@bison.howard.edu

***Abstract*—The Internet of Things (IoT) offers immense benefits by enabling devices to leverage networked resources thereby making intelligent decisions. The numerous heterogeneous connected devices that exist throughout the IoT system creates new security and privacy concerns. Some of these concerns can be overcome through trust, transparency, and integrity, which can be achieved with data provenance. Data provenance, also known as data lineage, provides a history of transformations that occurs on a data object from the time it was created to its current state. Data provenance has been explored in the areas of scientific computing, business, forensic analysis, and intrusion detection. Data provenance can help in detecting and mitigating malicious cyber attacks. In this paper, we explore the integration of provenance within the IoT. We introduce PAITS, a provenance collection framework for IoT devices. We evaluate the effectiveness of our framework by looking at an application of provenance data by developing a prototype system for proof of concept.**

I. Introduction

The Internet of Things (IoT) is transforming commercial and industrial information technology all over the world. IoT offers immense benefits in the areas of home, industrial and commercial automation however, the IoT is undergoing an exponential increase in the number of devices connected to the Internet. Accoring to a report released by cisco [1], It is estimated that over 50 million devices will be connected to the internet by the year 2020. With the vast amounts of connected heterogeneous devices, security and privacy risks are increased. For example, vulnerabilities in baby monitors allowed unauthorized access to devices whereby a malicious intruder can view live feeds from a remote location [2]. Most IoT devices are not initially developed with security in mind. This raises the complexity of including security after the device has been deployed. For instance some devices come with default passwords which might never be changed during the device lifecycle.

Due to the heterogeneity of devices and the level of uncertainty in deployments devices in IoT, trust is a critical step to ensuring the security of IoT devices. This can be achieved through data Provenance. Data provenance is a comprehensive history of activities that occurred on an entity from its origin to its current state. Provenance ensures confidence in the accuracy of data in an IoT system. Provenance has been applied in domains such as scientific workflow for experiment reproducibility, information security as a form of access control and also for intrusion detection in mitigating malicious adversaries. For IoT devices (things) that produce lots of sensor-actuator data, a workflow representation of how sensor data is generated can depict dependency between sensor readings and information in the device.

In this paper, we propose PAITS, a provenance collection framework for IoT systems, in which provenance data is collected and modeled to represent dependencies between sensor-actuator readings and the entities in an IoT system. Most of the interconnected heterogeneous devices (things) are embedded systems which require a lightweight and efficient solution as compared to general purpose systems. This requirement is attributed to the constrained memory and computing power of such. We also propose a provenance sensor model alignment which provides a direct correlation of sensor readings to provenance data.

The rest of the paper is organized as follows. In section II, we discuss background information on IoT and data provenance, we also motivate the need to incorporate provenance with IoT by using a use case scenario. In section III, we discuss how provenance model and sensor data are aligned. In section IV, we discuss implementation details for PAITS. In section V, we discuss related works. Finally, in section V, we conclude with future research directions.

II. Background

In this section, we describe key concepts of data provenance, IoT characteristics, and provenance models. We also provide a motivating example for the need for provenance collection via a use case.

A. Internet of Things

In 1991, Mark Weiser, a pioneer of ubiquitous computing envisioned the notion of computing interleaved in our daily lives [3]. Kevin Ashton, an early pioneer of IoT, defines the IoT as devices in our everyday lives which can be identified connected to a network [4]. These devices can learn from information gathered autonomously without human input which allows for improvements in waste reduction and overall standard of living. Buckley et al. [5] define the IoT as a network of billions of machines communicating with each other. Gubbi et al [6] defines the IoT as an interconnection of sensing and actuating devices that allows data sharing across platforms through a centralized framework. We define the

IoT as a network of heterogeneous devices with sensing and actuating capabilities communicating over the Internet.

IoT has applications in home automation, smart health, automotive communication, machine to machine communication, industrial automation. The interconnectivity between heterogeneous devices allows them to share information in a unique manner. Analytics is a driving force for IoT. With analytics, devices can learn from user data to make smarter decisions. IoT architecture represents a functional hierarchy of how information is disseminated from devices which contain sensing and actuating capabilities to massive data centers (cloud storage).

Figure 1 displays the IoT architecture and the interactions between the respective layers. IoT architecture consists of four distinct layers: sensor and actuator, device, gateway and cloud. The base of the architectural stack consists of sensors and actuators that gather information and interact with the device layer. The device layer consists of devices responsible for aggregating data collected from sensors and actuators. These devices in turn forward the aggregated data to the gateway layer. The gateway layer routes and forwards data collected from the device layer. The cloud layer is involved with the storage and processing of data collected from the gateway layer. Note that the resource constraints decrease up the architectural stack with the cloud layer having the most resources (memory, power, computation) and the sensor-actuator layer having the least.

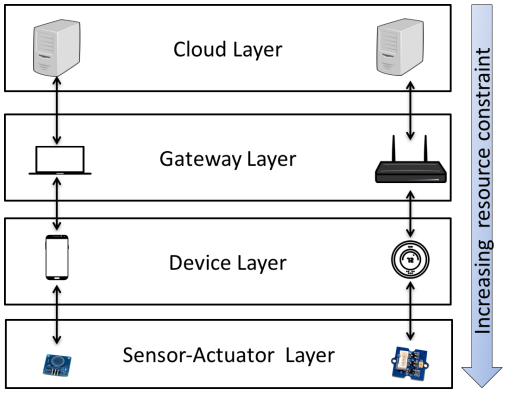

Fig. 1. IoT Architecture Diagram. The arrows illustrates the interaction between data at various layers on the architecture.

B. Provenance-Aware IoT Device Use Case

Creating a provenance-aware system is beneficial to IoT because it ensures the trust and integrity of interconn ected devices. Enabling provenance collection in IoT devices allows these devices to capture valuable information which enables backtracking in an event of a malicious attack.

Consider a smart home as illustrated in Figure 2 which contains interconnected devices such as a thermostat which automatically detects and regulates the temperature of a room based on prior information of a user's temperature preferences, a smart lock system that can be controlled remotely and

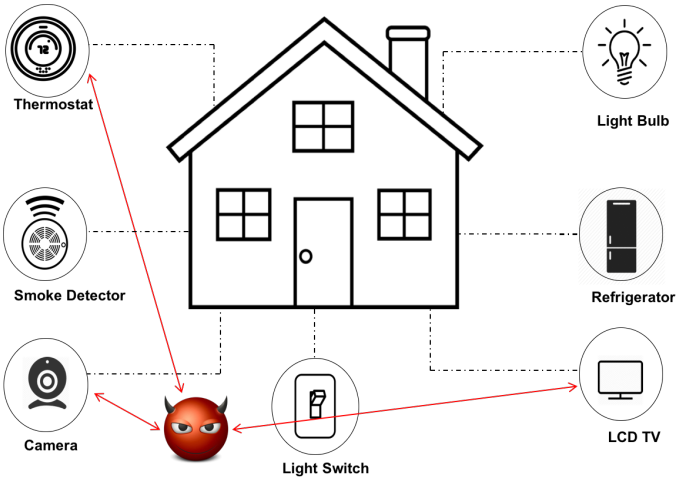

Fig. 2. Smart home use case scenario which demonstrates how provenance can be used to detect a point of entry of malicious intrusion

informs a user via short messaging when the door has been opened or is closed, a home security camera monitoring system, a smart fridge which sends a reminder when food products are low. In an event that a malicious intruder attempts to gain access to the smart lock system and security camera remotely, provenance information can be used to track the series of events to determine where and how a malicious attack originated. Provenance can also be used as a safeguard to alert of a possible remote or insider compromise thereby protecting against future or ongoing malicious attacks.

C. Data Provenance

The Oxford English dictionary defines provenance as "the place of origin or earliest known history of something" [7]. An example of provenance can be seen with a college transcript. A transcript is the provenance of a college degree because it outlines all of the courses satisfied in order to attain the degree. In the field of computing, data provenance, also known as data lineage, can be defined as the history of all activities performed on an entity from its creation to its current state. Cheney et al. [8] describes provenance as the origin and history of data from its lifecycle. Buneman et al [9] describes provenance from a database perspective as the origin of data and the steps in which it is derived in the database system. We formally define data provenance of an entity as a comprehensive history of activities that occur on that entity from its creation to its present state.

Provenance is best represented as an acyclic graph which denotes causal relationships and dependencies between entities.

1) Provenance Characteristics:

- Who: Provides information linking activities to an entity. Knowing the "who" characteristic is essential because it maps the identity of modification to a particular data object. An example of "who" in an IoT use case is a sensor device identifier.

- Where: Location information at which data transformation was made. This provenance characteristic could be optional since not every data modification contains location details. An example is a GPS coordinate.
- When: The time at which data transformation occurred. This is an essential provenance characteristic. Being able to tell the time of a data transformation allows for tracing data irregularities. An example of this characteristic is a timestamp that denotes when sensor data was read.
- What: The transformation applied. A use case for IoT can be seen in the various operations (create, read, update, and delete) that could be performed on a data object.

D. Model for Representing Provenance for IoT

Provenance of sensor readings in an IoT device should outline the dependency relationship between all entities responsible for producing those readings. Provenance can be represented using a modeling language with ontological representations such as PROV-DM which is represented serialized in three formats: XML, JSON and RDF. This model displays data dependency between agents and entities.

1) Provenance Data Model (Prov-DM): PROV-DM [10] is a model that conforms to PROV Ontology (PROV-O), a W3C standard used to depict dependencies between entities, activities and agents (digital or physical). PROV-DM creates a common model that allows for interchange of provenance information between heterogeneous devices.

PROV-DM contains two major components: types and relations. Types can be entities, activities, or agents. An entity is a physical or digital object. An activity represents some form of action that occurs over a time frame. An agent takes ownership of an entity, or performs an activity.

Figure 3 illustrates the types contained in PROV-DM and their representation. Entities, activities and agents are represented by oval, rectangle and hexagonal shapes respectively.

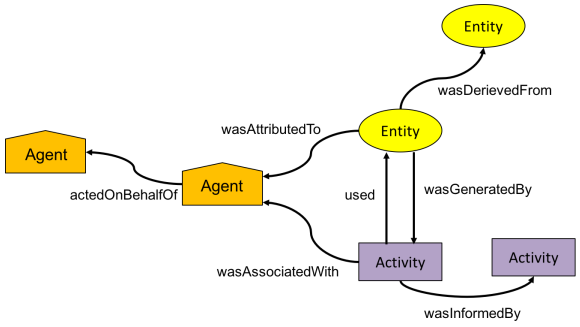

Fig. 3. Prov-DM respresentation showing various types contained in the model (Entity, Activity, and Agent) and the relationships between them

PROV-DM defines seven relationships between the types. These relationships are as follows

- wasGeneratedBy: Signifies the entity is completed by an activity
- used: An entity has been adopted by an activity in which one of the activity uses entity generated by another activity.
- wasInformedBy: The exchange of an entity by two activity in which one of the activity uses an entity generated by another activity.
- wasDerivedFrom: Represents a copy of information from an entity.
- wasAttributedTo: Denotes relational dependency between an entity and an agent when the activity that created the agent is unknown.
- wasAssociatedWith: An agent created or modified of the activity
- actedOnBehalfOf: Delegation of authority from an agent to itself or another agent to perform a particular responsibility.

III. Prov-Sensor Model Alignment

In this section, we introduce the sensor-provenance model mapping and explain how to use an ontology to convert sensor and actuator readings to provenance.

Sensor data contains observation information such as temperature, location details which can be transformed to various standardized data interchange formats (RDF, XML, JSON e.t.c). Trace data containing sensor-actuator readings are important components but do not depict dependency relationship when used alone. Trace data is transformed to provenance to represent causality and dependencies relationships between entities contained in an IoT system. Provenance Ontology and sensor data can be further aligned for better representation of dependency relationships between sensor data.

A single sensor might have the capability of collecting multiple trace data, td. For example, a sensor might be able to collect sensor readings of temperature, location, humidity. td at time t_1 can be defined as $td = \{x_1, y_1, z_1\}$ where x_1, y_1, z_1 are distinct sensor readings contained in the sensor at time t_1. A combination of trace data at a particular point in time is considered an event. We define an event for sensor s_1 as $e = \{td_1, ...td_n\}$ where td_1 is the first trace data collected by s_1 and td_n is the last.

$$[sensor, actuator, device] \in \{prov : agent\}$$
$$[datapoint] \in \{prov : entity\}$$

Using the Provenance Ontology, in an IoT system containing multiple sensors, device and sensor information, we represent device information as agents (prov:agents), we represent the operation performed on sensor-actuator readings (read, create, update) as a provenance activity (prov:activity). Events are represented as entities (prov:entity). We represent sensor trace as a tuple (t, e, a, s_1, r_1) where t represents a timestamp, e represents an event in which trace data is collected, a represents the action performed on sensor readings, s_1 represents sensor specific information such as sensor name, sensor version and r_1 represents device information such as device name, device version, and time of last update.

1) Example: Consider a humidity and temperature sensor s_1, connected to device r_1, a Raspberry Pi. Event, e, can be defined as $e = \{temperature, humidity\}$ therefore the tuple representation of trace data for Raspberry Pi r_1 with sensor s_1 taken at time t_1 can also be defined as $(t, \{temperature, humidity\}, a, s_1, r_1)$. Each tuple is mapped to the Provenance Ontology representation using the defined constructs in section IV. For each tuple, s_1, and r_1 are represented as entities as (prov:entity), a is represented as (prov:activity), and the e is represented as (prov:entities). Provenance is represented as a directed acyclic graph and the edge between two entities is considered a relation. Relations between the entities follows the Provenance Ontological convention which depicts transformation between entities. For example since s_1 is contained in device r_1, r_1 forms an edge with s_1 with the used relation. (e.g r_1 used s_1).

Figure 4 further illustrates the concept of Prov-Sensor alignment using a graphical representation which contains a device connected to three sensors s_1, s_2, and s_3 with events $[e_1, e_2, ..., e_n]$. Data is generated by three identical temperature sensors, s_1, s_2 and s_3. Trace data generated from devices are aligned with Provenance Ontology. The graph depicts data dependency between r_1, the three sensors, the activity performed by the sensors (the sensors create data in this case) and the events. Each event makes an edge with the preceding event. Edges between events are denoted with a dotted arrow . This represents time dependency between events. The graph of all of the events is:

$$G = (V, E)$$
$$V = e_1, ...e_n$$
$$E = (e_i, e_{i+1}) : e_i, e_{i+1} \in V, e_1 \neq e_{i+1}$$

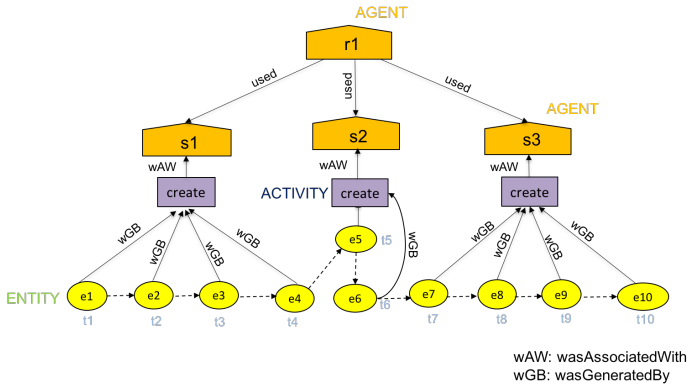

Fig. 4. Prov-Sensor Model Alignment

A. Algorithm Description

In Algorithm 1, F represents the file object containing tuples x. For each tuples contained in F, s, d represents sensor and device information respectively and are generated as agents. e is generated as events, a is generated as an activity. p is a memory representation of the provenance document containing all provenance types and their relations.

Algorithm 1: Prov-Sensor Alignment

```
Function trace2Prov (F)
    p ← createProvDocument()
    for x ∈ F do
        if s ∉ p then
            s ← createAgent()
        if d ∉ p then
            d ← createAgent()
        e ← createEntity()
        a ← createActivity()
        if x ∉ p then
            x ← relateSensorToDevice()
        if y ∉ p then
            y ← relateActivityToSensor()
        z ← relateEntityToActivity()
```

IV. PAITS System Implementation

In this section, we outline PAITS a trace-based provenance collection system for IoT devices. Figure 5 displays the system architecture. Sensor and actuator readings in the form of input and output (I/O) events are recorded by the tracer component. This component intercepts I/O and produces trace information represented in Common Trace Format (CTF).

PAITS converts CTF trace data to provenance in the PROV-DM IoT Model, which is described in Section IV. CTF conversion to PROV-DM is done using babeltrace. This conversion can happen at any layer of the IoT stack. Babeltrace is a plugin framework which allows the conversion of CTF traces into other formats. Trace or provenance data is securely transmitted to a gateway and later transmitted and stored in a cloud backend. Our backend of choice is Neo4j, a graph database with support for efficient storage, query and visualization of provenance data. CTF contains a mandatory stream known as metadata. Metadata contains information about other streams. It allows to parse a stream without specifying a layout.

1) Common Trace Format (CTF) : CTF encodes binary trace output information containing multiple streams of binary events such as I/O activity. CTF is composed of streams of events. Each event can be broken into as streams. Streams allow for fast processing since they do not have to be stored in disk before being sent over a network or processed in memory.

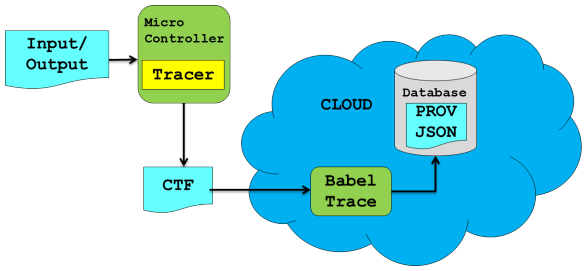

Fig. 5. System Architecture for PAITS.

Our goal is to create a provenance collection system which records I/O operations on data for devices connected in IoT. For our implementation, several tools and hardware components are used in the development of our prototype:

- Raspberry Pi is the microcontroller used to evaluate our approach. We chose Raspberry Pi because it is a representation of what can be found on an IoT gateway device. Raspberry Pi is a low cost, simple IoT demonstrator.
- Neo4j is a graph database which allows optimized querying of graph data such as provenance.
- Babeltrace is a trace converter tool to convert traces from one format into another.
- barectf is a light weight generator of C code for that collect data in CTF.
- A yaml generator in barectf creates yaml configuration files with information barectf needs to generate CTF trace output. Configuration files contain settings such as an application trace stream, packet type, payload type and size.

V. Related Work

Muniswamy-Reddy et al. [11] developed Provenance Aware Storage System (PASS), a provenance collection system that tracks system-level provenance of the Linux file system. Provenance information is stored in the same location as the file system for easy accessibility, backup, restoration, and data management. Provenance information is collected and stored in the kernel space. PASS is composed of 3 major components: provenance collector, provenance storage, and provenance query. The collector keeps track of system level provenance. It intercepts system calls which are translated into provenance data and initially stored in memory. Provenance data is then transferred to a file system in an kernel database, BerkleyDB which maps key value pairs for provenance data for fast index look up. Our approach looks at not just system level provenance but also application level provenance.

Bates et al. [12] developed, HiFi,a system level provenance collection framework for the Linux kernel using Linux Provenance Modules (LPM), which utilizes Linux Security Modules (LSM). LSM is a framework that was designed for providing custom access control inside the Linux kernel. HiFi contains three components: provenance collector, provenance log and provenance handler. The collector and log are contained in the kernel space while the handler is contained in the user space. The collector uses LSM to record provenance data and writes it to the provenance log. The handler reads the provenance record from the log. The log is a storage medium which transmits the provenance data to the user space. This approach to collecting provenance data differs from our work since we focus on memory constrained embedded systems which might not contain an operating system or a file system.

RecProv [13] is a provenance system which records user-level provenance, avoiding the overhead incurred by kernel level provenance recording. It does not require changes to the kernel like most provenance monitoring systems. It uses Mozilla rr to perform deterministic record and replay by monitoring system calls and non deterministic input. The provenance information generated is converted into PROV-JSON, and stored in Neo4j, a graph database for visualization and storage of provenance graphs.

Spillance et al. [14] developed a user space provenance collection system, Storybook that allows the use of application specific extensions such as database provenance, system level provenance, and web and email servers. Storybook captures provenance by intercepting system level events in the FUSE file system and storing in MySQL. StoryBook allows developers to implement provenance inspectors custom provenance models for specific applications which are often modified by different application (e.g web servers, databases). When an operation is performed on a data object, the appropriate provenance model is triggered and provenance data for that data object is captured. StoryBook stores provenance information such as open, close, read or write, application specific provenance, and causality relationship between entities contained in the provenance system. Provenance data is stored in key value pairs using Stasis and Berkely DB.

Lim et al. [15] developed a model for calculating the trust of nodes in a sensor network by using data provenance and data similarity as deciding factors to calculate trust. The value of provenance signifies that the more similar a data value is, the higher the trust score. Also, the more the provenance of similar values differ, the higher their trust score. The trust score of a system is affected by the trust score of the sensor that forwards data to the system. Provenance is determined by the path data travels through the sensor network. This work differs from our approach since the authors focus on creating a trust score and do not emphasize how the provenance data is collected.

VI. Conclusion and Future Work

In this paper, we motivate the need for integrating provenance into the IoT system. We propose PAITS, a provenance collection framework that provides provenance collection capabilities for devices in an IoT system. Some of the proposed future work are outlined below:

- Provenance collection raises privacy issues. How do we ensure that the vast amount of data collected is not invasive to privacy.
- Securing Provenance: Proper encryption and authentication techniques [16] are needed to ensure the confidentiality, and integrity of provenance data.

Acknowledgment

This research has been supported in part by US National Science Foundation (CNS grant No:1646317), and by a generous gift from Leidos. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF or Leidos.

REFERENCES

[1] D. Evans, "The internet of things how the next evolution of the internet is changing everything," https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, Apr 2011, accessed: 2016-10-01.

[2] M. Stanislav, "Hacking iot: A case study on baby monitor exposures and vulnerabilities," https://www.rapid7.com/docs/Hacking-IoT-A-Case-Study-on-Baby-Monitor-Exposures-and-Vulnerabilities.pdf, Sep 2015, accessed: 2016-10-01.

[3] M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.

[4] K. Ashton, "That internet of things thing," *RFiD Journal*, vol. 22, no. 7, pp. 97–114, 2009.

[5] L. Yan, Y. Zhang, L. T. Yang, and H. Ning, *The Internet of things: from RFID to the next-generation pervasive networked systems*. CRC Press, 2008.

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[7] Rationality., *The Cambridge Dictionary of Philosophy*. Cambridge University Press, 1999.

[8] J. Cheney, L. Chiticariu, and W.-C. Tan, "Provenance in Databases: Why, How, and Where," *Found. Trends databases*, vol. 1, no. 4, pp. 379–474, Apr. 2009. [Online]. Available: http://dx.doi.org/10.1561/1900000006

[9] P. Buneman, S. Khanna, and T. Wang-Chiew, "Why and Where: A Characterization of Data Provenance," in *Database Theory ICDT 2001*, ser. Lecture Notes in Computer Science, J. V. d. Bussche and V. Vianu, Eds. Springer Berlin Heidelberg, Jan. 2001, no. 1973, pp. 316–330, dOI: 10.1007/3-540-44503-X_20. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-44503-X_20

[10] "PROV-DM: The PROV Data Model," https://www.w3.org/TR/prov-dm/, accessed: 2016-10-01. [Online]. Available: https://www.w3.org/TR/prov-dm/

[11] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer, "Provenance-aware Storage Systems," in *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference*, ser. ATEC '06. Berkeley, CA, USA: USENIX Association, 2006, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267359.1267363

[12] D. J. Pohly, S. McLaughlin, P. McDaniel, and K. Butler, "Hi-Fi: Collecting High-fidelity Whole-system Provenance," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 259–268. [Online]. Available: http://doi.acm.org/10.1145/2420950.2420989

[13] Y. Ji, S. Lee, and W. Lee, "RecProv: Towards Provenance-Aware User Space Record and Replay," in *Provenance and Annotation of Data and Processes*, ser. Lecture Notes in Computer Science, M. Mattoso and B. Glavic, Eds. Springer International Publishing, Jun. 2016, no. 9672, pp. 3–15, dOI: 10.1007/978-3-319-40593-3_1. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-40593-3_1

[14] R. Spillane, R. Sears, C. Yalamanchili, S. Gaikwad, M. Chinni, and E. Zadok, "Story Book: An Efficient Extensible Provenance Framework," in *First Workshop on the Theory and Practice of Provenance*, ser. TAPP'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 11:1–11:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1525932.1525943

[15] H.-S. Lim, Y.-S. Moon, and E. Bertino, "Provenance-based Trustworthiness Assessment in Sensor Networks," in *Proceedings of the Seventh International Workshop on Data Management for Sensor Networks*, ser. DMSN '10. New York, NY, USA: ACM, 2010, pp. 2–7. [Online]. Available: http://doi.acm.org/10.1145/1858158.1858162

[16] R. Hasan, R. Sion, and M. Winslett, "The case of the fake picasso: Preventing history forgery with secure provenance," in *Proccedings of the 7th Conference on File and Storage Technologies*, ser. FAST '09. Berkeley, CA, USA: USENIX Association, 2009, pp. 1–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1525908.1525909